

Final Degree Project

**Bachelor's Degree in Industrial Technology Engineering**

**Adaptive Cruise Control, an scaled model:  
Platooning using two-wheeled robots**

**REPORT**

**Author:** Carlos Conejo Barceló  
**Director:** Arnau Dòria Cerezo  
Víctor Repecho del Corral  
**Call:** June, 2017



Escola Tècnica Superior  
d'Enginyeria Industrial de Barcelona



## Summary

The aim of the project is to recreate, in a small-scale, the interaction between vehicles that circulate in the same path. Several technological options like Adaptive Cruise Control or Cooperative Adaptive Cruise Control are studied and compared to allow a safety circulation, avoiding accidents between vehicles or between vehicles and pedestrians.

This Final Degree Project is formed by three differentiated parts. The first task consists on triplicating a two-wheel robot, which was made by Bachelor's Degree students one year ago, to have four equal vehicles. The second step is to study the behaviour of the different robots and its sensors, and try to make a statistical study to reduce these possible deviations unifying the code that vehicles rule. Finally, the last part and the principal one, consists on programming a platooning position control and the connection between the different vehicles and the computer.

# Contents

<b>SUMMARY</b>	<b>1</b>
<b>CONTENTS</b>	<b>2</b>
<b>1. GLOSSARY</b>	<b>4</b>
<b>2. PREFACE</b>	<b>5</b>
2.1. Source.....	5
2.2. Motivation.....	5
2.3. Previous requirements .....	5
<b>3. INTRODUCTION</b>	<b>6</b>
3.1. Objectives .....	6
3.2. Vehicles construction .....	6
<b>4. VEHICLE CHARACTERIZATION</b>	<b>7</b>
4.1. Line Sensor.....	8
4.2. HC-SR04 Ultrasonic Sensor .....	9
4.3. DC Motors.....	9
4.4. Solution .....	10
<b>5. POSITION CONTROLLER'S DESIGN</b>	<b>11</b>
5.1. Introduction .....	11
5.1.1. Control Problem .....	12
5.2. Adaptive Cruise Control (ACC) .....	14
5.2.1. Introduction .....	14
5.2.2. Proportional Action in Adaptive Cruise Control.....	14
5.2.3. Proportional-Integral Action in Adaptive Cruise Control.....	15
5.3. Cooperative Adaptive Cruise Control (CACC) .....	15
5.3.1. Introduction .....	15
5.3.2. Proportional Action in Cooperative Adaptive Cruise Control .....	16
5.3.3. Proportional-Integral Action in Cooperative Adaptive Cruise Control .....	16
5.4. Simulations .....	16
5.4.1. Proportional Action in ACC and CACC .....	17
5.4.2. Proportional-Integral Action in ACC and CACC .....	18
5.5. Signal Filters .....	20
5.6. Selected Controller and Implementation.....	22
5.7. Experimental Results .....	23
5.7.1. Break Testing.....	23
5.7.2. Circulation Testing .....	25

<b>6. VEHICLE'S CONNECTIVITY</b>	<b>27</b>
<b>7. ANNEX</b>	<b>28</b>
7.1. ESP8266 Software Upgrade.....	28
7.1.1. Preparation Hardware: .....	28
7.1.2. PuTTY Terminal.....	29
7.1.3. Firmware Installation (ESP flash download tool v.2.3).....	29
7.2. Firmware launched in PCB .....	31
7.2.1. Get MAC address Function.....	31
7.2.2. Characterize Vehicle function.....	32
7.2.3. IP Address assignation function .....	33
7.2.4. Position Controller function .....	34
7.3. Circuits design.....	35
<b>CONCLUSIONS</b>	<b>36</b>
<b>ACKNOWLEDGEMENTS</b>	<b>37</b>
<b>BIBLIOGRAPHY</b>	<b>38</b>
<b>TECHNICAL CONCEPTS</b>	<b>39</b>

# 1. Glossary

First, it is important to know which computer software has been used during the project and its application.

**Eclipse:** Software platform formed by programming tools and used for developing codes in free programming languages. Used to write firmware codification and debug it into the electronic board. See more information at <https://eclipse.org/>

**MATLAB:** Mathematical software used to solve all kind of operations. It has its own programming language. Its application consists on solving numerical problems like long iterations and it allows to plot data obtained from simulations.

**Simulink:** MATLAB's subprogram that has the benefit of having the possibility of simulating all kind of electrical or electronical circuits. For more information about MATLAB and Simulink, visit [https://es.mathworks.com/?s\\_tid=gn\\_logo](https://es.mathworks.com/?s_tid=gn_logo)

**PuTTY:** Software that allows connection between PC and the wireless module by AT commands. It is also used to upgrade module's software. To download this program visit <http://www.putty.org/>

**Python:** Software language and platform, based in other older languages, which utilization is easier for the user. Used to transfer data between PC and vehicles. See more information in <https://www.python.org/>

## 2. Preface

### 2.1. Source

In the last years, society's interests have changed considerably. As it is known, it has been a quick evolution in interaction between people. In the past, the chances of communication were limited to talking or sending letters. Nowadays, there is a new tendency that consists on being connected whenever and wherever with everybody, using old methods or social networks.

Technology is helping to develop our world in a comfortable and a safety way. Connectivity is one of the most important tools to make society's demands real.

### 2.2. Motivation

The decision of carrying out this project is due to the mix of interests about new technologies applied in automotive sector and robotics.

Companies like Tesla Motors, Mercedes or BMW are improving their products by adding components that allow an autonomous drive and are developing connectivity systems with the objective of making drive easier, safer and trying to save energy or combustible when it is possible. This is a reason to study, in a small-scale, how controllers are designed and implemented to achieve these goals.

### 2.3. Previous requirements

There was only one 2-wheeled robot completely constructed. Therefore, three more vehicles weeded to be assembled before starting the mainly part of the project.

This project is based in three previous degree's final projects, which were made last year. Two students wrote the first and the second one at the same time. A third student, relying on these two mentioned projects, extended them implementing new ways to achieve the global objective, which consists on creating an autonomous-drive system for "handmade" vehicles.

- Control design and implementation for a line tracker vehicle (Prats, 2016): The author designed and constructed the basic parts of one vehicle (hardware), and programmed a firmware<sup>1</sup> that allowed the 2-wheeled robot to follow paths (marked by lines).
- Disseny i implementació d'un Sistema de comunicacions Wi-Fi per a una xarxa de vehicles autònoms (Riera, 2016): The student created local network to connect 2-wheeled robots to a personal computer with the objective of controlling remotely a vehicle and to study its behaviour.
- Design of controllers and its implementation for a line tracker vehicle (Costa, 2017): The author took charge of putting together the projects mentioned before and implemented a system to control the motion of the 2-wheeled robot over lines.

## 3. Introduction

### 3.1. Objectives

As it is explained in the preface, in the previous projects there were designed controllers to allow an autonomous drive with total independence of other cars or interferences with the environment. The aim of the project is to make an interaction between vehicles, which are in the same circuit.

To be more precise, the most important objectives to carry out are:

- Find an easy way to characterize every single vehicle: It is important to know that 2-wheeled robots have the same sensors but it exists an important variability between them and, consequently, it could induce to measure errors.
- Design a standardized velocity control: Vehicles must adapt their celerity to obstacles that can find between their paths.

### 3.2. Vehicles construction

Before starting to focus on the objectives, as it is mentioned in the preface, it is necessary to construct more 2-wheeled robots to prove if the upgrades are really working.

All the elements needed to assemble each vehicle have been bought at Leantec (Leantec) as a kit named *Kit Robot LRE-EO2*. These are the components included in every kit:

- Chassis robot *2WD*.
- Battery holder.
- *L298N* motor driver<sup>2</sup>. (STMicroelectronics, 2000)
- USB cable to send the firmware from the PC to the board.
- *HC-SR04* Ultrasonic sensor. (ElecFreaks.com)
- Coloured wires.

Other essential elements are not incorporated in the mentioned kit:

- *STM32F4-Discovery*. (STMicroelectronics, 2017)
- *LM317T* regulator to get 3V from the 5V contributed by the batteries. (STMicroelectronics, 2014)
- *ESP8266* Wi-Fi module. (Espressif, 2017)
- *LRE-F22* sensor (Line Sensor).

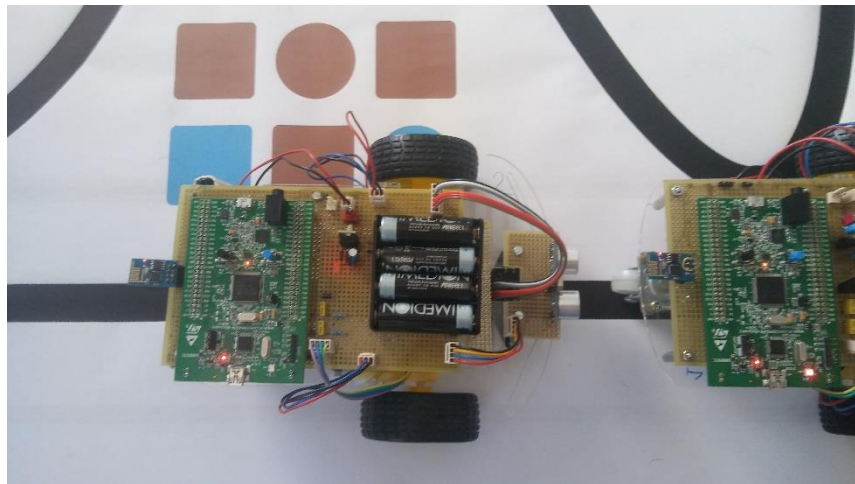
As it has been explained before, there was a 2-wheeled robot assembled the year before. Consequently, it has been made the decision of imitating the construction once to prove all network connections between vehicles. At the same time that this project is being carried out, a co-worker is trying to optimize the hardware by designing a new method to construct the two remaining 2-wheeled robots.

## 4. Vehicle Characterization

Testing vehicles with the same firmware inside (made for *Vehicle 1*), it is shown that every car do not respond in the same way with exactly the same conditions. Consequently, this fact could induce errors in the global system. To avoid them it has been decided to characterize vehicles in function of their own properties.

The firmware contains, at the start, a characterization code (see Annex 7.2.2.), which is dedicated to identify the vehicle and attribute its particular equations that define its performance. With the purpose of facilitating the characterization work, every car is identified by a number on the chassis.

- 1: Vehicle constructed before. The sensor identification values have been studied in previous projects.
- 2, 3, 4: In spite of being assembled with the same sensors as in *Vehicle 1*, it is necessary to analyse the behaviour of the captors in every car.



(Figure 4-1) Vehicle 1 on the right side of the image and vehicle 2 on the left. As it is shown, both have an identification number on its chassis.

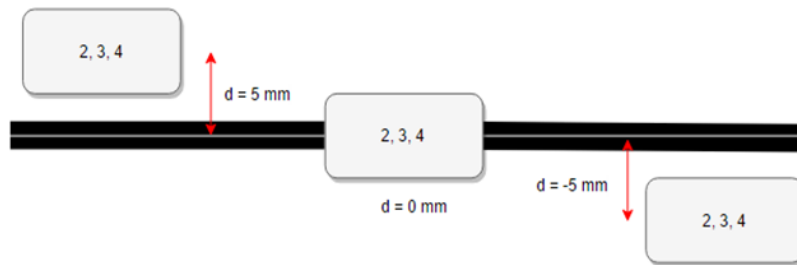


## 4.1. Line Sensor

The equation, which describes the deviation of the vehicle from the line, is:

$$d_{measured} = m \cdot V_{diff} + n \quad (4.1)$$

- $V_{diff}$ : Difference between the voltage proportioned by FL1 and FL2 sensors, soldered in the inferior surface of the line sensor.
- $d_{measured}$ : Horizontal distance in  $mm$  between the centre of the line and the real position of the car. To find  $m$  and  $n$  parameters, it is used the same values of this variable in all the experiments ( $-5\text{ mm}$ ,  $0\text{ mm}$ ,  $5\text{ mm}$ ).



(Figure 4-2)  
Representation of  
the experiment  
carried out in the  
laboratory to find  
 $m$  and  $n$   
constants).

By experimentation, it is measured by *Eclipse* the value of the variable  $V_{diff}$  in every different vehicle (Table 4-1). With this data, it is possible to obtain  $m$  and  $n$  of (4.1).

$V_{diff}$ (V)	-5 mm	0 mm	5 mm
Vehicle 2	2755	-135	-3025
Vehicle 3	2091	-383	-2860
Vehicle 4	2455	-126	-2710

(Table 4-1) Line Sensor measures obtained in the laboratory.

Vehicle	1	2	3	4
$m$	0,00173	0,00202	0,0019574	0,001937
$n$	0,234	0,775	0,1109	0,2453

(Table 4-2)  $m$  and  $n$  parameters obtained interpolating data showed in (Table 4-1).

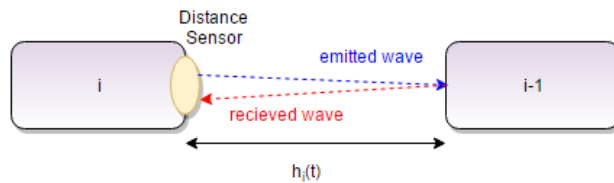
## 4.2. HC-SR04 Ultrasonic Sensor

The measures proportioned by the distance sensor were identic in every vehicle while having the same conditions. It is not necessary to characterize each car with different parameters. According to the HC-SR04 Ultrasonic Sensor datasheet, the value sent by the captor is a time between the emission and the reception of the ultrasonic signal<sup>3</sup>, which is proportional to the distance between the car and the closest object  $h_i(t)$ .

The behaviour is showed in (Figure 4-2) and follows the next equation:

$$h_i(t) = \text{high level time} \cdot \frac{340}{2} \text{ m/s}$$

- $h_i(t)$ : Empirical Distance between the studied vehicle and his predecessor.
- *High-level time*: period of the ultrasonic wave transmission.



(Figure 4-3) Ultrasonic Sensor working representation.

The celerity of the ultrasonic waves is the same as the speed of sound,  $340 \text{ m/s}$ . This value is divided by 2 because it has to be considered that the distance between the car and the object is travelled twice.

## 4.3. DC Motors

Like all hardware components that constitute the 2-Wheel Robot, DC motors<sup>4</sup> also have variability in their behaviour. That is the reason why it has been studied the reaction of each motor to different particular voltages separately. The results of the angular velocity of the wheels, obtained by a python application thanks to measures of the encoders, were considerably similar in all motors.

As a result, it is considered that all DC motors have the same equations as the ones used in vehicle 1 (studied in previous projects).

## 4.4. Solution

Once the vehicles are characterized with their own properties, it is important to think about how is the firmware able to give every different identity to the cars.

Considering the intention of implementing the same code in all vehicles, the solution contributed consists on finding a property that allows sorting out cars with the security of not having any errors.

This distinctive property is the Media Access Control (MAC) address<sup>5</sup>, unique for every Wi-Fi device. The question is how the firmware asks the MAC address to each device. It is important to get this direction at the start to classify the cars and avoid motion problems at the beginning of the vehicle's motion.

The communication between the PCB<sup>6</sup> and the Wi-Fi device is set through the serial port (UART<sup>7</sup>). See Annex 7.2.1. The command "AT+CIPSTAMAC?" takes charge of asking the MAC address. The buffer's structure is shown in (Table 4-3).

49	56	58	102	101	58	51	52	58	57	98	58	99	55	58	53	52
'1'	'8'	':'	'f'	'e'	':'	'3'	'4'	':'	'9'	'b'	':'	'c'	'7'	':'	'5'	'4'
49	56	58	102	101	58	51	52	58	57	99	58	56	54	58	51	56
'1'	'8'	':'	'f'	'e'	':'	'3'	'4'	':'	'9'	'c'	':'	'8'	'6'	':'	'3'	'8'
49	56	58	102	101	58	51	52	58	57	98	58	99	52	58	51	100
'1'	'8'	':'	'f'	'e'	':'	'3'	'4'	':'	'9'	'b'	':'	'c'	'4'	':'	'3'	'd'
49	56	58	102	101	58	51	52	58	57	98	58	99	50	58	54	50
'1'	'8'	':'	'f'	'e'	':'	'3'	'4'	':'	'9'	'b'	':'	'c'	'2'	':'	'6'	'2'

(Table 4-3) This table shows the MAC direction of every different vehicle (represented with different colours each and ordered from 1<sup>st</sup> to 4<sup>th</sup>). The elements on the top of each colour are the decimal MAC address and the ones on the bottom are the hexadecimal MAC address.

## 5. Position Controller's Design

### 5.1. Introduction

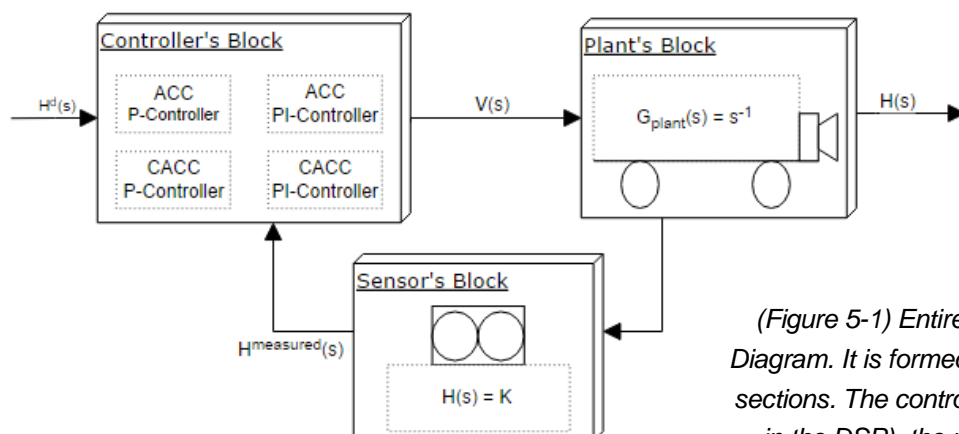
This section contains the most important part of the entire project. The structure followed to arrive at an optimum solution of the problem presented at the start, has consisted on performing the steps of the Scientific Method: *Present the problem, ask questions, background research, construct hypothesis, test the hypothesis, analyse data and make conclusions*.

In this case, the objective is to design a controller with the objective of regulating the position of a vehicle depending on the environmental conditions. The system's structure is shown in the following block diagram<sup>8</sup>:

To achieve this goal, firstly it is important to study which variables have to be modified, *output variables*, and which ones have to be received or treated, *input variables*. Once this step is clear, it is necessary to study how to turn the input variables into output variables by following a *transfer function*<sup>9</sup>, different in every case presented.

As a position controller, *input and output variables* of the complete system must be position variables. The entire system is structured by the following parts or blocks, and showed in (Figure 5-1).

- Plant's Block, which is estimated with a pure integrator. Consequently, the input has to be a speed variable.
- Controller's Block that takes charge of transforming a position into a speed variable. This transform process is studied in this part of the project.
- Sensor's Block, which can be implemented with a constant. It allows to have a real value to adjust position thanks to the controller.

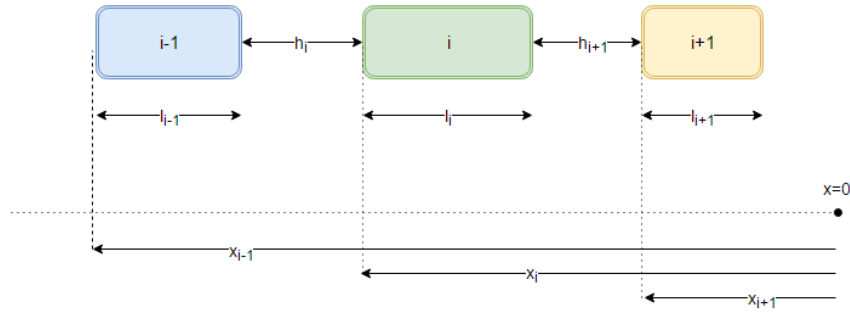


(Figure 5-1) Entire System Blocks Diagram. It is formed by three different sections. The controller (implemented in the DSP), the plant (estimated experimentally) and the ultrasonic sensor available in the vehicle.

### 5.1.1. Control Problem

Firstly, it is important to define clearly the variables and constants which are necessary to define the problem. They are shown in (Figure 5-2) and explained just forward:

- $i$ : Reference number of the studied vehicle. It is bounded to  $1 \leq i \leq N$ . Where  $N$  is the total number of vehicles in the same path.
- $x_i(t)$ : Instantaneous position of *vehicle*  $i$ .
- $x_{i-1}(t)$ : Instantaneous position of *vehicle*  $i - 1$ .
- $h_0(t)$ : Security distance between cars when they are stopped. Configured by the user, considering a safety criteria established experimentally.
- $h_i(t)$ : Empirical distance between positions of *vehicle*  $i$  and  $i - 1$ . Measured by the ultrasonic sensor.
- $l_i$ : *vehicle*  $i$ 's length. In this project, all lengths are equal due to the cloning method followed.



(Figure 5-2) Vehicle circulation performance and graphical explication of variables and parameters.

It has been considered that  $N$  vehicles are circulating along the same path. The distance of *vehicle*  $i$  ( $i = 1, \dots, N$ ) with respect to its predecessor, is given by:

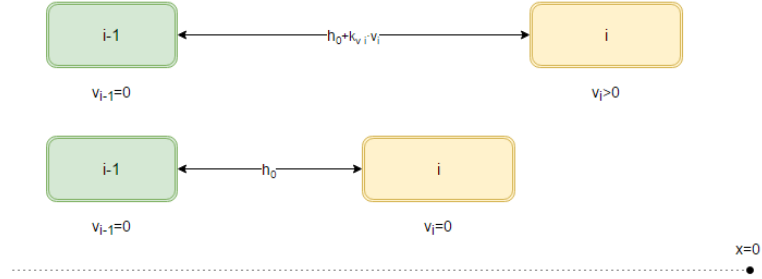
$$h_i(t) = x_{i-1}(t) - x_i(t) - l_{i-1} \quad (5.1)$$

The aim of the adaptive cruise control algorithm is to regulate the distance,  $h_i$  to a desired value:

$$h_{i\text{ref}}(t) = h_{i0}(t) + v_i(t) \cdot k_{iv} \quad (5.2)$$

Where  $h_{i0}$  is the standstill distance of *vehicle*  $i$  (available from measurements) and  $k_{iv}$  is the constant time headway (equivalent to the time that the *vehicle*  $i$  takes to arrive at the position of its predecessor).

The reference distance is a function of the vehicle's instant velocity. When the speed grows, the distance has to be increased for safety reasons (to avoid accidents between vehicles). This fact is shown in (Figure 5-3).



(Figure 5-3) Behaviour's description of security distance depending on the speed of the vehicle.

For sake of simplicity, it is considered that:

$$h_{i0}(t) = h_0 \quad k_{iv} = k_v$$

The error function, which has to be minimized by the controller, is represented by:

$$e_i(t) = h_{i\text{ref}}(t) - h_i(t) \quad (5.3)$$

Differentiating with respect to the time (5.1) and (5.2), vehicle's speed equations are obtained:

$$\frac{dh_i(t)}{dt} = v_{i-1}(t) - v_i(t)$$

$$\frac{dh_{i\text{ref}}(t)}{dt} = k_v \cdot \frac{dv_i(t)}{dt}$$

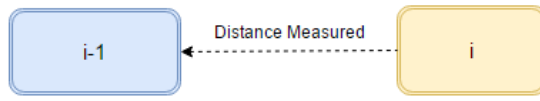
To summarize, the objective of designing controllers is to modify the car's speed depending on an input of the distance between a vehicle ( $i$ ) and its predecessor ( $i - 1$ ). This value is given by the ultrasonic sensor and has to be regulated to achieve the goal of having a distance value as close as possible to the reference distance estimated.

## 5.2. Adaptive Cruise Control (ACC)

### 5.2.1. Introduction

Adaptive Cruise Control, also called Autonomous Cruise Control is a safety technology that consists on regulating the speed of a vehicle depending on the distance between itself and its predecessor. It is graphically explained on (Figure 5-4).

In this project, the measures of distance will be given by the distance sensor. The following step will be to calculate the velocity needed with the controller designed. Finally, this value will be given to the DC motors to achieve the desired speed.



(Figure 5-4) Structure of a vehicle's ACC System.

As it has been mentioned in (5.3), the error function is referred to the difference between the reference and the real distance that a vehicle has with its predecessor.

According to (Dòria-Cerezo, 2017) the error dynamics in the Adaptive Cruise Control, obtained deriving the error function (5.3) and considering a null predecessor's speed, has the following form:

$$\dot{e}_i = k_v \cdot \dot{v}_i + v_i \quad (5.4)$$

### 5.2.2. Proportional Action in Adaptive Cruise Control

As it has been explained in the ideal controller, to guarantee closed loops dynamics in a proportional controller, it is needed:

$$\dot{e}_i = -k_p \cdot e_i \quad (5.5)$$

Finally, starting from the error dynamics' equation (5.4) and replacing the error's derivate with a product of a proportional constant and the error (5.5), it is possible to describe controller's behaviour by following the next equation:

$$k_p \cdot (h_i - h_0) = (1 + k_p \cdot k_v) \cdot v_i + k_v \cdot \dot{v}_i$$

Applying Laplace Transform, where  $s$  is the time-operator and  $h_i$  the value measured by the distance sensor, the continuous-time controller's equation is:

$$V_i(s) = \frac{k_p}{k_v \cdot s + (1 + k_p \cdot k_v)} \cdot (H_i(s) - H_0(s))$$

### 5.2.3. Proportional-Integral Action in Adaptive Cruise Control

In this project, vehicles do not know local variables, such as linear speed, distance measured by sensors ... from others. It does not exist communication between vehicles directly. That is the reason why it has considered that in this controller  $v_{i-1}$  is unknown. To make this possible, it is necessary to implement a Proportional-Integral Controller:

$$\dot{e}_i = -k_p \cdot e_i - k_z \cdot z_i \quad (5.6)$$

$$\dot{z}_i = e_i \quad (5.7)$$

Making the same process as in proportional action, that consists on equalizing equation (5.4) with equations (5.6) and (5.7) the new controller's equation is obtained:

$$(1 + k_p \cdot k_v) \cdot v_i + k_v \cdot \dot{v}_i = -k_p \cdot (h_0 - h_i) - k_z \cdot z_i$$

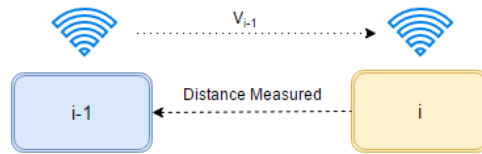
Applying Laplace Transform, the transfer function obtained does not depend on any predecessor's variable.

$$V_i(s) = \frac{(k_p \cdot s + k_z)}{k_v \cdot s^2 + (1 + k_p \cdot k_v) \cdot s + k_z \cdot k_v} \cdot (H_i(s) - H_0(s))$$

## 5.3. Cooperative Adaptive Cruise Control (CACC)

### 5.3.1. Introduction

Cooperative Adaptive Cruise Control includes the ACC technology, adding speed and acceleration measures from the predecessor vehicle. It provides a bigger sense of safety due to the system behaviour to the changes in the preceding vehicle speed. The response got is quicker than in ACC. (Figure 5-5) shows how this systems work.



(Figure 5-5) Structure of a Vehicle's CACC System

According to (Dòria-Cerezo, 2017), error dynamics in a Cooperative Adaptive Cruise Control, obtained deriving the error function and considering that there is a connection between vehicles that allows the knowledge of the vehicle's predecessor speed, has the following form:

$$\dot{e}_i = k_v \cdot \dot{v}_i - v_{i-1} + v_i \quad (5.8)$$



### 5.3.2. Proportional Action in Cooperative Adaptive Cruise Control

The design of the controller would be very similar as the one made for the ACC control. The only difference is the integration of the predecessor's velocity inside the transfer function (substituting equation (5.4) for (5.8) in all process):

$$k_p \cdot (h_i - h_0) + v_{i-1} = (1 + k_p \cdot k_v) \cdot v_i + k_v \cdot \dot{v}_i$$

Applying Laplace Transform, the continuous-time controller's equation is:

$$V_i(s) = \frac{k_p}{k_v \cdot s + (1 + k_p \cdot k_v)} \cdot (H_i(s) - H_0(s)) + \frac{1}{k_v \cdot s + (1 + k_p \cdot k_v)} \cdot V_{i-1}(s)$$

Where  $s$  is the time-operator,  $v_{i-1}$  is the speed of the predecessor vehicle, considered while having a cooperative system, and  $h_i$  the value measured by the distance sensor.

### 5.3.3. Proportional-Integral Action in Cooperative Adaptive Cruise Control

Like in Proportional design, the controller would be very close to the created in the ACC control, with the difference of including the predecessor's speed inside the transfer function:

$$(1 + k_p \cdot k_v) \cdot v_i + k_v \cdot \dot{v}_i - v_{i-1} = -k_p \cdot (h_0 - h_i) - k_z \cdot z_i$$

The continuous-time transfer function obtained by Laplace transforming,

$$V_i(s) = \frac{(k_p \cdot s + k_z)}{k_v \cdot s^2 + (1 + k_p \cdot k_v) \cdot s + k_z \cdot k_v} \cdot (H_i(s) - H_0(s)) + \frac{s}{k_v \cdot s^2 + (1 + k_p \cdot k_v) \cdot s + k_z \cdot k_v} \cdot V_{i-1}(s)$$

## 5.4. Simulations

To achieve the objective of choosing the optimum speed controller for the vehicles, it is important to proceed firstly to a response behaviour's analysis by simulating it using *MATLAB* program.

The simulation is also compared with the real response of the 2-wheeled robot. To stablish communication between vehicle and computer (where data can be plotted), Wi-Fi module and Python program will be used. The first element has the function of sending all data required via Wi-Fi. The second takes charge of obtaining this data (allowing to watch it in real time or with a little delay) and saving it in a document, that enables the option of plotting it in *MATLAB* to see if there is a considerable difference between simulation and real response.

### 5.4.1. Proportional Action in ACC and CACC

The characteristic Proportional-Controller equation is followed by:

$$\dot{e} + k_p \cdot e = 0$$

Applying Laplace transform,

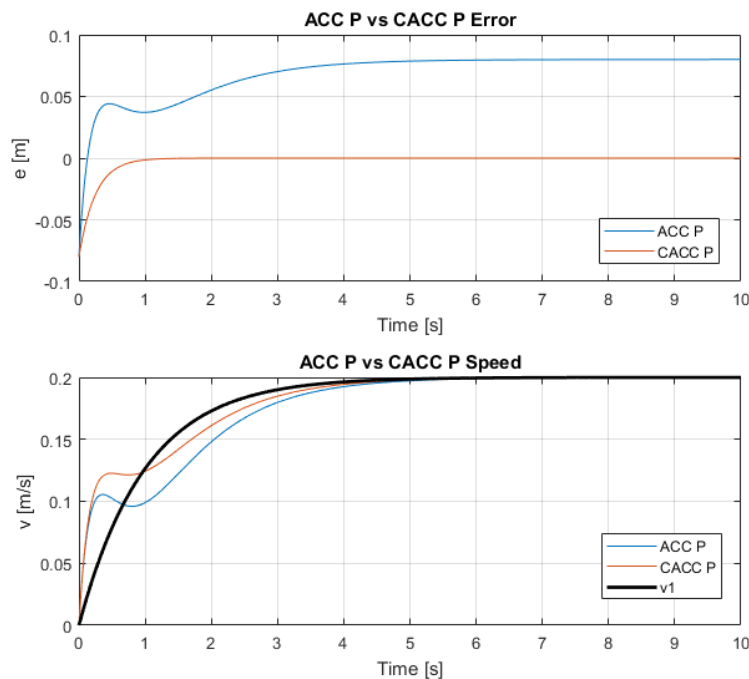
$$E(s) \cdot (s + k_p) = 0 \quad (5.9)$$

The next step is to study the stability: Stability condition in continuous-time: The real part of the pole must be negative.  $\text{Re}\{s\} < 0$ .

$s = -k_p$  is the pole of equation (5.9). Consequently,  $k_p > 0$  guarantees asymptotic stability with a time constant:  $\tau = \frac{1}{k_p}$

This defined constant is proportional to the 2% settling time in a first order system,  $t_s = 4 \cdot \tau$

Error in stationary state is different in both cases. In CACC systems, the error has a tendency to approximate to zero in the settling time. On the other hand, ACC system has a stationary error shown in (Figure 5-6).



(Figure 5-6) Graph, which describes the error and the speed comparing them in proportional ACC and CACC cases. The settling time considered is 1 second.

### 5.4.2. Proportional-Integral Action in ACC and CACC

Mixing both error dynamics' equations explained before, it has been found that the following equation has to be accomplished in every instant of time. The system's characteristic equation:

$$\ddot{e} + k_p \cdot \dot{e} + k_z \cdot e = 0$$

$$E(s) \cdot (s^2 + k_p \cdot s + k_z) = 0 \quad (5.10)$$

Once the controller's function is obtained, it is necessary to define  $k_p$  and  $k_z$  constants to get stability in the designed system. It can be observed that it is a second-order function with poles of equation (5.10) in:

$$s_{1,2} = -\frac{k_p \pm \sqrt{k_p^2 - 4 \cdot k_z}}{2}$$

The next step will be to study the stability: Stability condition in continuous-time: The real part of the pole must be negative.  $Re\{s_{1,2}\} < 0$ .

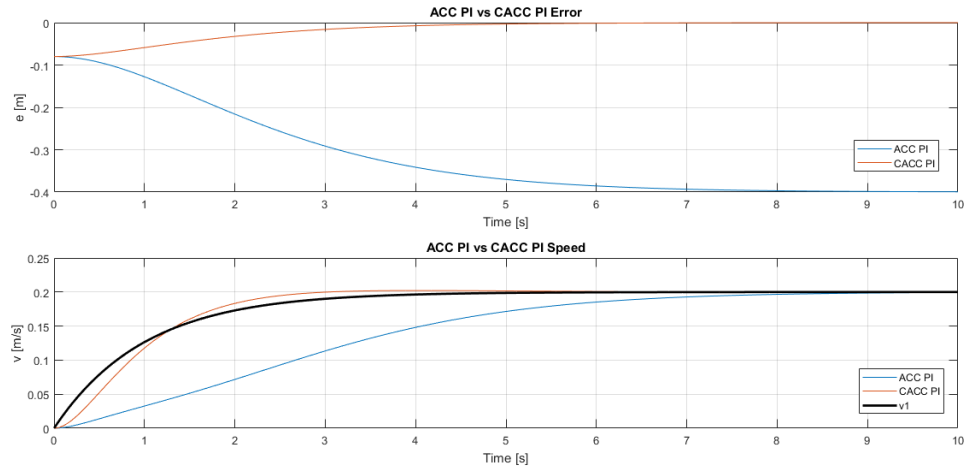
- **Critically damped Response:  $k_p^2 = 4 \cdot k_z$**

Poles, with this condition, are:  $s_{1,2} = -\frac{k_p}{2}$

Consequently, it is needed  $k_z, k_p > 0$  to be stable

The critically damped response solution is shown in (Figure 5-7) and has the form:

$$e(t) = (c_1 + c_2 \cdot t) \cdot e^{-\frac{k_p \cdot t}{2 \cdot k_z}}$$



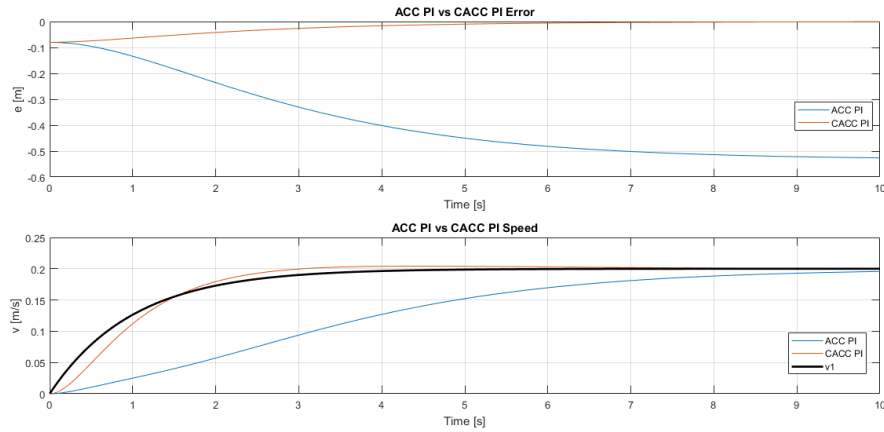
(Figure 5-7) Error and speed solution when the response is critically damped. In this case, the simulation has been performed with  $k_p = 2$  and  $k_z = 1$ .

- **Overdamped Response:  $k_p^2 > 4 \cdot k_z$**

Poles, following the condition showed before, are:  $s_{1,2} = -\frac{k_p \pm \sqrt{k_p^2 - 4 \cdot k_z}}{2}$

Therefore, to have stability in the system, it is necessary to have  $k_z, k_p > 0$ .

The general solution is showed in (Figure 5-8) and is ruled by:  $e(t) = c_1 \cdot e^{s_1 \cdot t} + c_2 \cdot e^{s_2 \cdot t}$



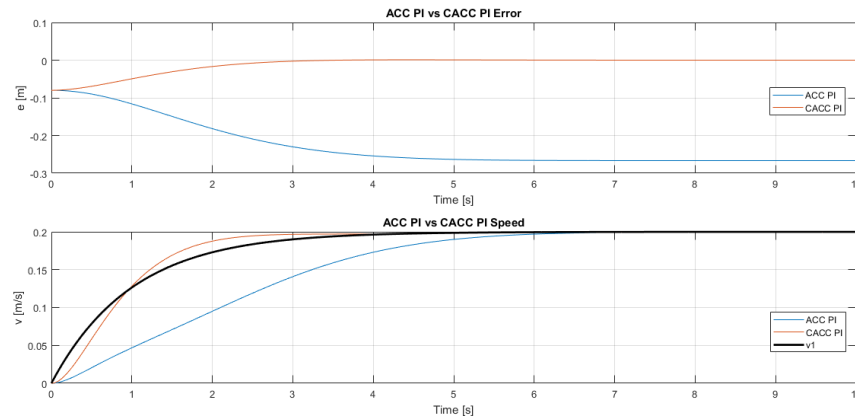
(Figure 5-8) Error and speed solution when the response is overdamped. In this case, the simulation has been performed with  $k_p = 2$  and  $k_z = 0.75$ .

- **Underdamped Response:  $k_p^2 < 4 \cdot k_z$ :**

Poles found with the last condition are:  $s_{1,2} = -\frac{k_p \pm j \sqrt{4 \cdot k_z - k_p^2}}{2}$

Consequently, stability conditions are:  $k_z, k_p > 0$

General Solution is showed in (Figure 5-9):  $e(t) = c_1 \cdot e^{\sigma \cdot t} \cdot \cos(w_p \cdot t) + c_2 \cdot e^{\sigma \cdot t} \cdot \sin(w_p \cdot t)$



(Figure 5-9) Error and speed solution when the response is underdamped. In this case, the simulation has been performed with  $k_p = 2$  and  $k_z = 1.5$ .

Simulation parameters are the same in all cases, the only value changed is the integrative constant  $k_z$ . From all graphics extracted from *MATLAB*, the conclusion is that all CACC systems work properly and have the tendency of reducing the error until zero. On the other hand, ACC systems have an error in stationary state. To minimize this error, it has been shown that it is necessary to implement an underdamped system.

Mathematically, this error in stationary state can be calculated from the Final Value Theorem:

First, it is important to find the error transfer equation that depends on the estimated predecessor's vehicle speed and the real one.

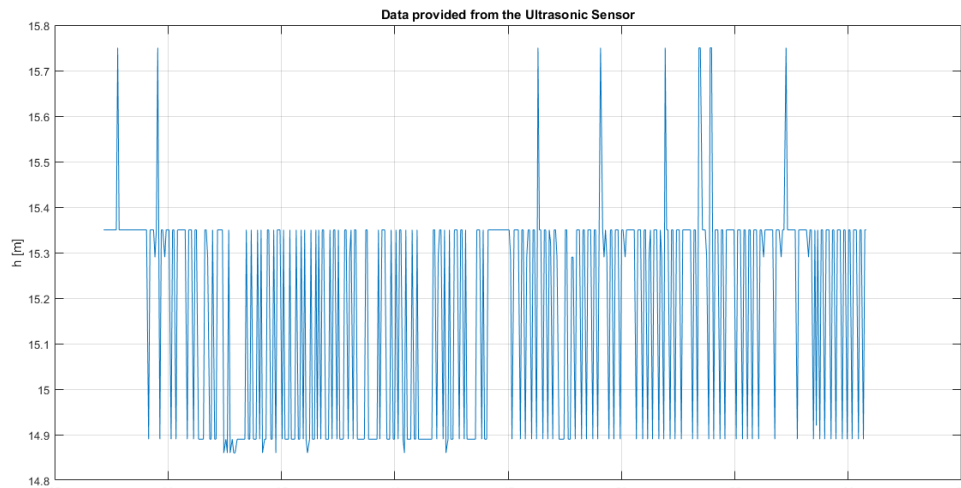
$$E(s) = \frac{k_v \cdot s^2 + (1 + k_p \cdot k_v) \cdot s + k_p}{(1 + k_v + k_p \cdot k_z) \cdot s^2 + (k_v \cdot k_z + k_p) \cdot s + k_z} \cdot (v_{i-1}^{Real} - \hat{v}_{i-1}) \quad (5.11)$$

Then, Final Value Theorem is applied to equation (5.11) to find the error value in stationary time.

$$\lim_{t \rightarrow \infty} e(t) = \lim_{s \rightarrow 0} s \cdot E(s) = \frac{k_p}{k_z} \cdot (v_{i-1}^{Real} - \hat{v}_{i-1})$$

## 5.5. Signal Filters

Because of the important dependence observed before between the data acquired by the Ultrasonic Sensor ( $h_i$ ), which is a controller's input variable, and the speed command given (output variable) to the 2-Wheel Robot, it is important to study if it exists an important variability on this purchase. In (Figure 5-10) is shown the behaviour of the sensor while it is completely stopped. It seems like data moves around two values with a similar frequency every time.



(Figure 5-10) Ultrasonic Sensor measures graph while the vehicle stands in a constant distance from an object of 15mm.

Due to the variability observed on the Ultrasonic Sensor acquired data (*Figure 5-10*), it has been decided to implement a low-pass filter with the objective of reducing this measures' error.

$$G_{filter}(s) = \frac{f_0}{s + f_0}$$

Where  $f_0$  is the maximum frequency of the variation of data that is accepted.

From (*Figure 5-10*) it has been obtained that  $f_{measures} = 10Hz$ , with a repeatability of values every 200ms.

It is known that Ultrasonic Sensor works automatically in  $f_{sensor} = 40Hz$  (sampling time is 25ms), so it has been considered that measures behaviour is similar as the sensor's. That is why repeatability has stabilised on a period of 50ms:

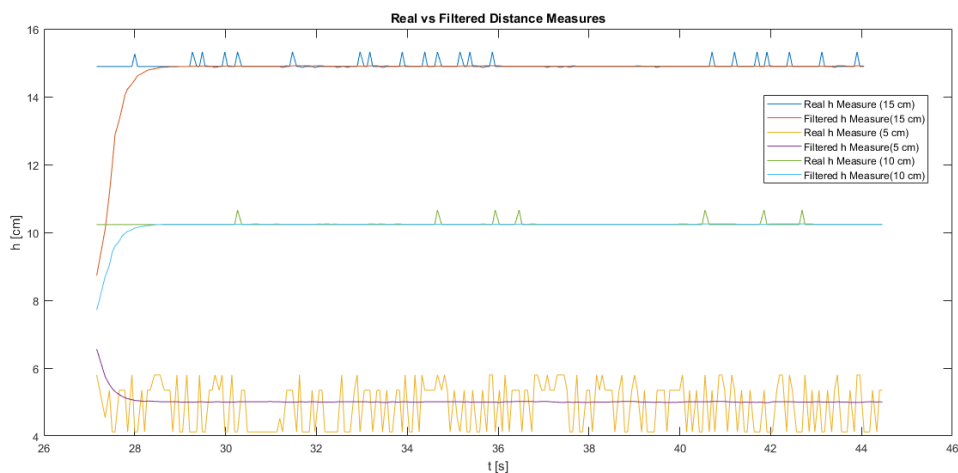
$$f_{real} = \frac{1}{50ms} = 20Hz$$

$$10 \cdot f_0 \leq f_{real}$$

$$f_0 = 2Hz$$

The filter's transfer function implemented rules the following structure, and is showed working on (*Figure 5-11*):

$$G_{filter}(s) = \frac{1}{0.5 \cdot s + 1}$$



(*Figure 5-11*) Comparison between real distance and filtered measures (5mm, 10mm and 15mm).

## 5.6. Selected Controller and Implementation

Comparing all controller's studied in the project, it has been observed that if the objective is to minimize error in stationary time, it is necessary to implement a CACC control system, where error tends to zero.

Due to experimental limitations, the studied vehicle cannot know predecessor's speed. In this case, CACC cannot be implemented, and options are reduced to P and PI ACC controllers. Finally, as a consequence of the advantages that PI-Controllers have over P-Controllers, it has been decided to implement the ACC PI-Controller with an underdamped response, which minimizes error's final value.

The next step is to allow the transfer's function implementation. To achieve the goal, it is necessary to convert this function mentioned from continuous to discrete-time. It is important to consider that the DSP board cannot work in continuous-time.

Firstly, it is necessary to group variables in each side of the equality,

$$[k_v \cdot s^2 + (1 + k_p \cdot k_v) \cdot s + (k_p + k_z \cdot k_v)] \cdot V(s) = (k_z + k_p \cdot s) \cdot (H_i(s) - H_0(s))$$

The next step is to convert the Continuous-Time function in  $s$  into a differential equation,

$$\frac{dv^2}{dt^2} + \frac{1 + k_p \cdot k_v}{k_v} \cdot \frac{dv}{dt} + \frac{k_p + k_z \cdot k_v}{k_v} \cdot v = \frac{k_z}{k_v} \cdot (h_i - h_0) + \frac{k_p}{k_v} \cdot \frac{dh_i}{dt}$$

To facilitate the visualization of discretization task, there has been created three parameters, which depend on the selected controller constants:  $a_1$ ,  $a_0$  and  $b_0$ .

$$\frac{dv^2}{dt^2} + a_1 \cdot \frac{dv}{dt} + a_0 \cdot v = b_0 \cdot (h_i - h_0) + b_1 \cdot \frac{dh_i}{dt} \quad a_1 = \frac{1 + k_p \cdot k_v}{k_v}, a_0 = \frac{k_p + k_z \cdot k_v}{k_v}, b_0 = \frac{k_z}{k_v}, b_1 = \frac{k_p}{k_v}$$

An easy way to discretize this differential equation is to create two related variables and find its dependence on discrete-time,

$$x_1 = v \tag{5.12}$$

$$x_2 = \frac{dv}{dt} \tag{5.13}$$

$$x_3 = \frac{dh_i}{dt}$$

Differentiating equations (5.12) and (5.13) the from respect to the time,

$$\dot{x}_1 = \dot{v} = x_2 \tag{5.14}$$

$$\dot{x}_2 = \ddot{v} = -a_1 \cdot x_2 - a_0 \cdot x_1 + b_0 \cdot (h_i - h_0) + b_1 \cdot x_3 \tag{5.15}$$

Finally, approximating the derivation of (5.14) and (5.15) into a difference of two consecutive measures in a sampling time division, it has been found:

$$v[k] = x_1[k] = x_2[k-1] \cdot T_s + x_1[k-1]$$

$$x_2[k] = T_s \cdot (-a_1 \cdot x_2[k-1] - a_0 \cdot x_1[k-1] + b_0 \cdot (h_i[k-1] - h_0) + b_1 \cdot x_3[k-1]) + x_2[k-1]$$

$$x_3[k] = \frac{h_i[k] - h_i[k-1]}{T_s}$$

Discrete-time controller has been included in firmware. See its codification in Annex 7.2.4.

## 5.7. Experimental Results

To make sure that the controller's implementation has been made properly, it has been decided to compare simulation (MATLAB) with real (Python) data.

To get this objective, the initial conditions in both states must be equal and the behaviour should be considerably similar.

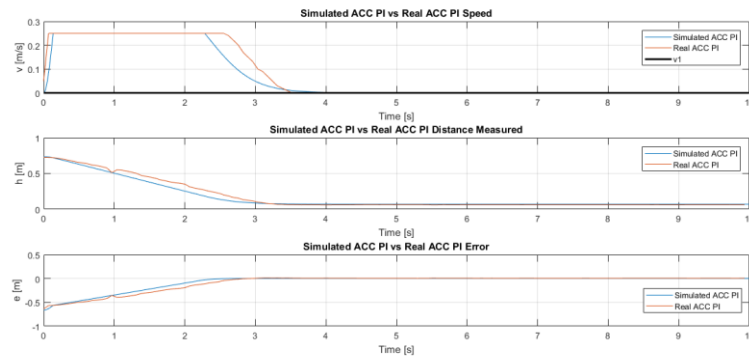
There have been made two different experiments in the circuit shown in the Annex 7.3.: break testing and circulation testing, everyone with repetitions changing little details that are explained in every figure.

### 5.7.1. Break Testing

The first experiment consists on placing a vehicle completely stopped in a specific distance from the studied vehicle. See the behaviour of the vehicle on (Figure 5-12) and (Figure 5-13).

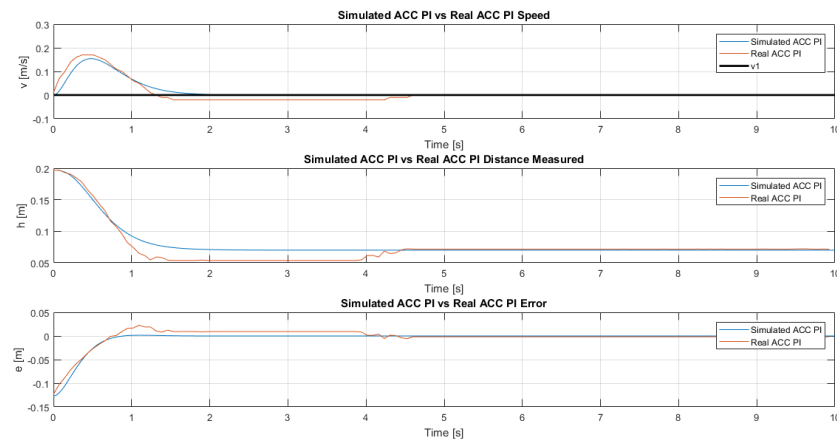
In both figures, vehicle's and simulated errors tend to the same value, zero, due to the equality of the estimated and the real value of the predecessor's speed, zero. Analysing both graphs can be observed that real distance measured and real error curves have the same shape as simulated ones. However, it exists a delay associated to friction and inertial problems, which have not been considered during the implementation of the controller.





(Figure 5-12) Plot of the speed, the distance measured by the Ultrasonic sensor and the error provided by the 2-wheeled robot, comparing it with simulations, in definite instants of time. In this case, the studied vehicle is initially 70cm away from its predecessor, which is stopped ( $v_1 = 0$ ).

In (Figure 5-12) all plotted curves seems to have a delay. The shape is nearly the same, but this mentioned error induces a non-equivalence of behaviour.

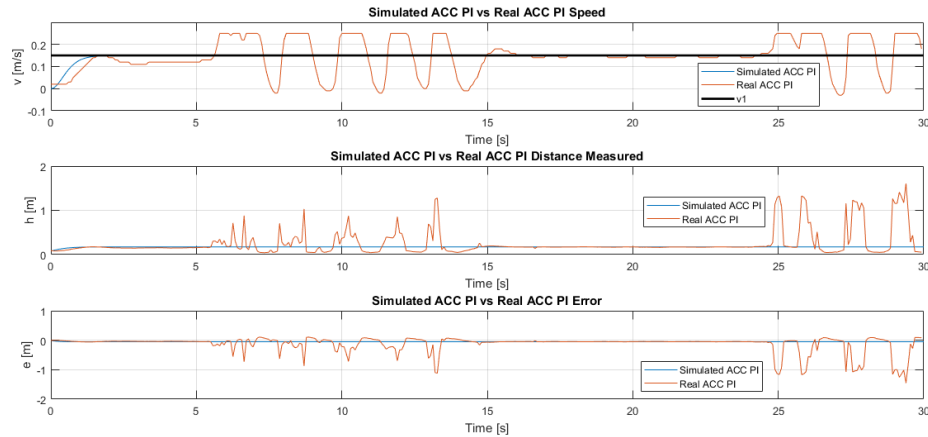


(Figure 5-13) Plot of the speed, the distance measured by the Ultrasonic sensor and the error provided by the car, comparing it with simulations, in definite instants of time. In this case, the studied vehicle is initially 20cm away from its predecessor, which is stopped ( $v_1 = 0$ ).

In (Figure 5-13) is shown how the vehicle approximates too much into its predecessor. To correct this “failure”, the instant tendering is to reduce this induced error, proportioning a negative speed to motors.

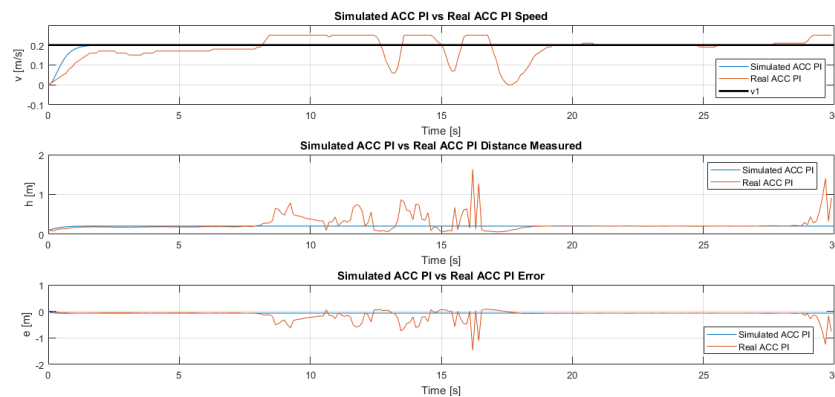
### 5.7.2. Circulation Testing

The second experiment consists on switching on two vehicles, initially the one forward with a constant velocity and in a specific distance from the studied vehicle. See the behaviour of the vehicle on (Figure 5-14) and (Figure 5-15)



(Figure 5-144) Plot of the speed, the distance measured by the Ultrasonic sensor and the error provided by the car, comparing it with simulations, in definite instants of time. In this case, the studied vehicle is initially 10cm away from its predecessor, which is stopped at the start but has the tendency of circulating at its maximum speed rate ( $v_1 = 0.2$ ).

In the first case of circulation testing (Figure 5-14), it can be observed how the vehicle studied, in  $t \in (0s, 5s)$ , tends to achieve the maximum speed value of the predecessor to stabilize error. Later,  $t \in (5s, 15s)$ , sensor, error and speed values oscillate due to circuit's curvature that causes vision problems from the studied vehicle to its predecessor.



(Figure 5-15) Plot of the speed, the distance measured by the Ultrasonic sensor and the error provided by the car, comparing it with simulations, in definite instants of time. In this case, the studied vehicle is initially far away from its predecessor, which is stopped at the start but has the tendency of circulating at its maximum speed rate ( $v_1 = 0.15$ ).

In this second case of circulation testing (*Figure 5-15*), it can be seen that vehicle studied, initially,  $t \in (0s, 8s)$ , tends to regulate its speed with the predecessor's one to stablish the error in the straight section. Later,  $t \in (8s, 17s)$ , sensor, error and speed values oscillate due to circuit's curvature that causes vision problems from the studied to its predecessor.

As a close circuit, vehicle's behaviour is the same in every cycle.

## 6. Vehicle's Connectivity

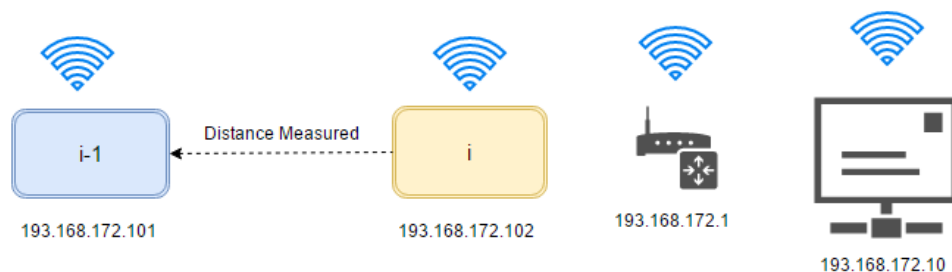
Connectivity structure, shown in (Figure 6-1), is changed to have the possibility of studying  $N$  vehicles at the same time.

Firstly, is important to know the rolls that components of the network make in our connectivity system.

- Wireless Router: Distributes commands that arrive from vehicles or from computer.
- Personal Computer: Used for plotting all data received from vehicles and it gives the possibility of changing parameters from vehicles in an easy way.
- *Vehicle i*: Receives commands from the computer and sends data of interest for analysing its behaviour.

To achieve the goal of having a solid connectivity structure, the first step consists on allocating different IP addresses<sup>10</sup> to every element of the network that covers all connections. See (Figure 6-1).

- Wireless Router IP: 193.168.172.1
- Personal Computer IP: 193.168.172.10
- *Vehicle 1* IP: 193.168.172.101
- *Vehicle 2* IP: 193.168.172.101
- *Vehicle N* IP: 193.168.172.10 $N$



(Figure 6-1) Connectivity Network, which takes charge of data distribution.

To assign *vehicle i*'s IP address, it is necessary to send a "AT+CIPSTA" command to the Wi-Fi module indicating the IP wanted. Codification is showed and explained in the Annex 7.2.3.

The Wireless Router restricts the possible simultaneous connections. It is important to know this maximum value of devices to avoid connectivity problems.

## 7. Annex

### 7.1. ESP8266 Software Upgrade

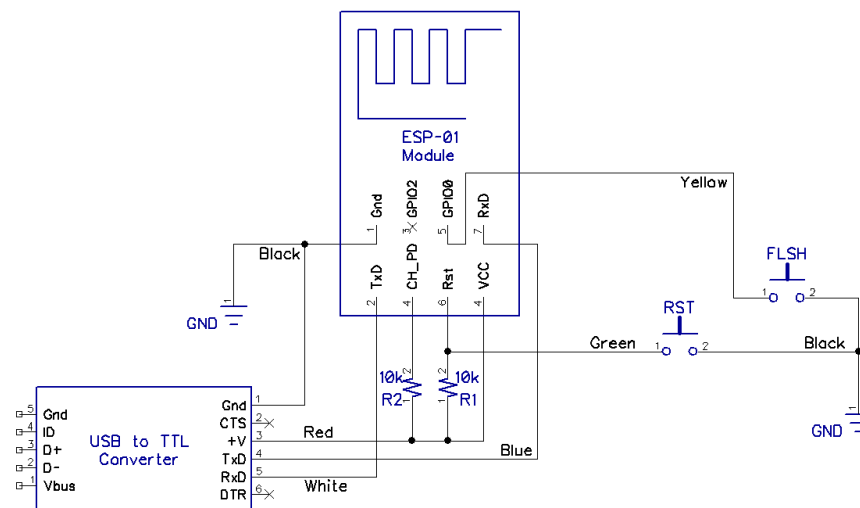
Due to several problems had with the software proportioned by the company, it has been decided to install the 1.4.1 firmware version. This upgrade allows communicating quicker (baud rate<sup>11</sup> of 115200 bits per second) and by AT commands<sup>12</sup>, used to enable the network that includes vehicles and computer.

It is important to follow the instructions of the following website: <https://www.allaboutcircuits.com/projects/update-the-firmware-in-your-esp8266-wi-fi-module/>. This link helps to follow systematically the upgrade. Because of the inconveniences found when setting up the firmware, it has been explained by detail how to prepare the installation for the ESP8266 available in the laboratory.

#### 7.1.1. Preparation Hardware:

With the objective of facilitating the installation, a particular board has been assembled.

The circuit implemented fulfils (Figure 7-1):

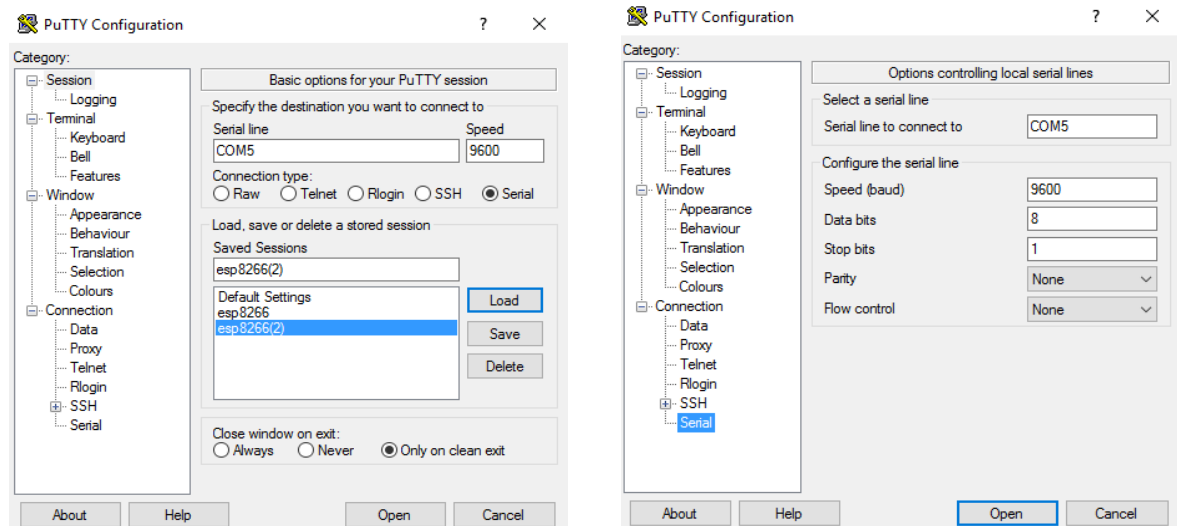


ESP-01 Connection Diagram

(Figure 7-1) Electronical circuit needed to set up the upgrade in the ESP8266 Module.

### 7.1.2. PuTTY Terminal

Download PuTTY on the following link: <http://www.putty.org/>. Once the program is set up, it has to be configured as it is indicated in the figure. COM port depends on the assignation given by PC.



Clicking on “open”, a command window is opened, and the communication between the PC and the ESP8266 is allowed. It is important to press “enter” followed by “ctrl” + “j” to send AT commands to the Wi-Fi module.

### 7.1.3. Firmware Installation (ESP flash download tool v.2.3)

The website where the program which allows the installation can be downloaded, is: <http://bbs.espressif.com/viewtopic.php?f=57&t=433>.

The following step is to download the version 1.4.0 and 1.4.1 files. First of all, go to the link <http://bbs.espressif.com>, click on the “SDKs” entry under Downloads, and then click on "latest release" under Announcements. The latest release of the Non-OS SDK (Software Development Kit) is what you want, and it would seem that you could click just under "Latest Version: 1.4.0" and get the latest version. But that's not quite right; notice that there is a patch available identified as esp\_iot\_sdk\_v1.4.1\_15\_10\_22. It's not really a patch; it's a corrected version of version 1.4.0. You want that and the AT\_v0.50 bin files. Click on each of those in turn and download the files; you can save them wherever you want, but they are fine in the Downloads folder.

## 6.1. Without Support For Cloud Update (FOTA)

### 1. 512KB Flash

bin	Address	Description
<b>master_device_key.bin</b>	0x3E000	Obtained from Espressif Cloud by users themselves to get Espressif Cloud service
<b>esp_init_data_default.bin</b>	0x7C000	Stores default RF parameter values, provided in SDK
<b>blank.bin</b>	0x7E000	Stores default system parameter values, provided in SDK
<b>eagle.flash.bin</b>	0x00000	Compiled by the steps said above
<b>eagle.irom0text.bin</b>	0x40000	Compiled by the steps said above

The first bin package is not used during the process. “Esp\_init\_data\_default.bin” and “blank.bin” are included in the 1.4.1 version directory. On the other hand, both following bin files are located in the 1.4.0 software version directory.

## 7.2. Firmware launched in PCB

### 7.2.1. Get MAC address Function

```
//Initialization of variables

int vehicle;
uint8_t MAC_address[40]={0};
uint8_t MAC[17]={0};
int i;
int j;

//Get MAC address from WiFi module

void get_MAC()
{
    HAL_Delay(1000);
    uint8_t comandaATE[] = "ATE0\r\n"; //Command used to initialize communication between PCB
    and WiFi module
    enviaUart(&comandaATE, COUNTOF(comandaATE) -1); //Sends the command to the WiFi module
    HAL_Delay(1000);
    inicialitzaBufferFIFO(&bufferEntradaUART); //Creates a buffer to allow an understandable
    communication
    uint8_t comandaCIPSTAMAC[] = "AT+CIPSTAMAC?\r\n"; //Command used to ask the MAC address
    enviaUart(&comandaCIPSTAMAC, COUNTOF(comandaCIPSTAMAC) -1); //Sends the command
    HAL_UART_Receive(&UartHandle,&MAC_address,100,10000); //Receives the MAC address and stores
    it in "MAC_address"
    int i=13; //Experimentally it is known that the MAC address starts in the 13th position and
    finishes in the 29th
    int j=0;

    //Copy from 13th until 29th string contained in "MAC_address" to "MAC"

    while (i<=29)
    {
        MAC[j]=MAC_address[i];
        i=i+1;
        j=j+1;
    }

    // Depending on the MAC address of the vehicle, associates the variable car

    if (MAC[15]==53 && MAC[16]==52)
    {
        vehicle=1;
    }
    else if (MAC[15]==51 && MAC[16]==56)
    {
        vehicle=2;
    }
    else if (MAC[15]==51 && MAC[16]==100)
    {
        vehicle=3;
    }
    else if (MAC[15]==54 && MAC[16]==50)
    {
        vehicle=4;
    }

    i=13;
    j=0;
    characterize_vehicle(vehicle);
}
```



## 7.2.2. Characterize Vehicle function

```
// Initialize new constants
// Line sensor equation constants
float m;
float n;

//Filter constants
float Ts_sensor=0.025;
float Tf=0.5;
float a_filter;

// Position controller constants
float kp=2;
float kz=1.5;
float kv=0.35;
float a1;
float a0;
float b0;

// Set the maximum speed that each vehicle can achieve
float v_max;

void characterize_vehicle(vehicle)
{
    a_filter=Ts_sensor/(Ts_sensor+Tf);

    a1=(1+kp*kv)/kv;
    a0=(kp+kz*kv)/kv;
    b0=(kz/kv);

    if (vehicle==1)
    {
        m=0.00173;
        n=0.234;
        v_max=0.25;
    }
    else if (vehicle==2)
    {
        m=0.0019805;
        n=0.4423;
        v_max=0.2;
    }
    else if (vehicle==3)
    {
        m=0.0019574;
        n=0.1109;
        v_max=0.15;
    }
    else if (vehicle==4)
    {
        m=0.001937;
        n=0.2453;
        v_max=0.1;
    }
}
```

### 7.2.3. IP Address assignation function

```

int configurarConnexionWifi(vehicle)
{
    HAL_Delay(500);
    uint8_t comandaATE[] = "ATE0\r\n"; // Echo deactivator command
    enviaUart(&comandaATE, COUNTOF(comandaATE) -1);
    HAL_Delay(500);
    inicialitzaBufferFIFO(&bufferEntradaUART);
    uint8_t comandaCWMODE[] = "AT+CWMODE=1\r\n"; // Multi-connectivity mode command
    enviaUart(&comandaCWMODE, COUNTOF(comandaCWMODE) -1);
    HAL_Delay(500);

    // Send IP Address allocation to the vehicle
    if (vehicle==1)
    {
        uint8_t comandaCIPSTA[] =
"AT+CIPSTA=\"192.168.173.101\", \"192.168.173.1\", \"255.255.255.0\" \r\n";
        enviaUart(&comandaCIPSTA, COUNTOF(comandaCIPSTA) -1);
    }
    else if (vehicle==2)
    {
        uint8_t comandaCIPSTA[] =
"AT+CIPSTA=\"192.168.173.102\", \"192.168.173.1\", \"255.255.255.0\" \r\n";
        enviaUart(&comandaCIPSTA, COUNTOF(comandaCIPSTA) -1);
    }
    else if (vehicle==3)
    {
        uint8_t comandaCIPSTA[] =
"AT+CIPSTA=\"192.168.173.103\", \"192.168.173.1\", \"255.255.255.0\" \r\n";
        enviaUart(&comandaCIPSTA, COUNTOF(comandaCIPSTA) -1);
    }
    else if (vehicle==4)
    {
        uint8_t comandaCIPSTA[] =
"AT+CIPSTA=\"192.168.173.104\", \"192.168.173.1\", \"255.255.255.0\" \r\n";
        enviaUart(&comandaCIPSTA, COUNTOF(comandaCIPSTA) -1);
    }

    else
    {
        uint8_t comandaCIPSTA[] =
"AT+CIPSTA=\"192.168.173.110\", \"192.168.173.1\", \"255.255.255.0\" \r\n";
        enviaUart(&comandaCIPSTA, COUNTOF(comandaCIPSTA) -1);
    }

    HAL_Delay(500);

    // Set connection between WiFi module and router

    uint8_t comandaCWJAP[] = "AT+CWJAP=\"wireless\", \"xarxaesp8266\" \r\n";
    enviaUart(&comandaCWJAP, COUNTOF(comandaCWJAP) -1);
    HAL_Delay(7000);
    return(0);
}

```

### 7.2.4. Position Controller function

```
//Initialization of controller variables & constants

float h_real;
float h_previous=0.07;
float h0=0.07;
float h;
float v;
float x1;
float x1_previous=0;
float x2;
float x2_previous=0;
float x3;
float x3_previous=0;
float u1;
float Ts=0.00005;

// Discrete-time controller implementation

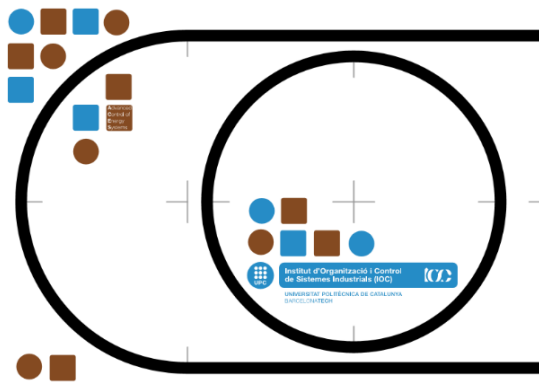
void Position_Controller()
{
    h_real=(float)USd; // Assigns the real distance measured (USd)
    h=h_previous*(1-a_filter)+a_filter*h_real;
    x1=x2_previous*Ts+x1_previous;
    x2=Ts*(-a1*x2_previous-a0*x1_previous+b0*(h_previous-h0)+b1*x3_previous)+x2_previous;
    x3=(h-h_previous)/Ts;
    x1=v;
    x1_previous=x1;
    x2_previous=x2;
    x3_previous=x3;
    u1=v;

    // Speed Saturation

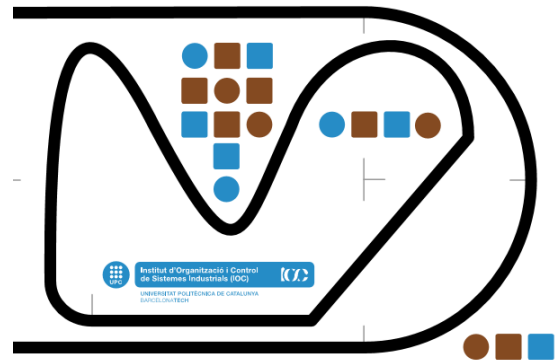
    if (v>v_max)
    {
        u1=v_max;
    }
}
```

### 7.3. Circuits design

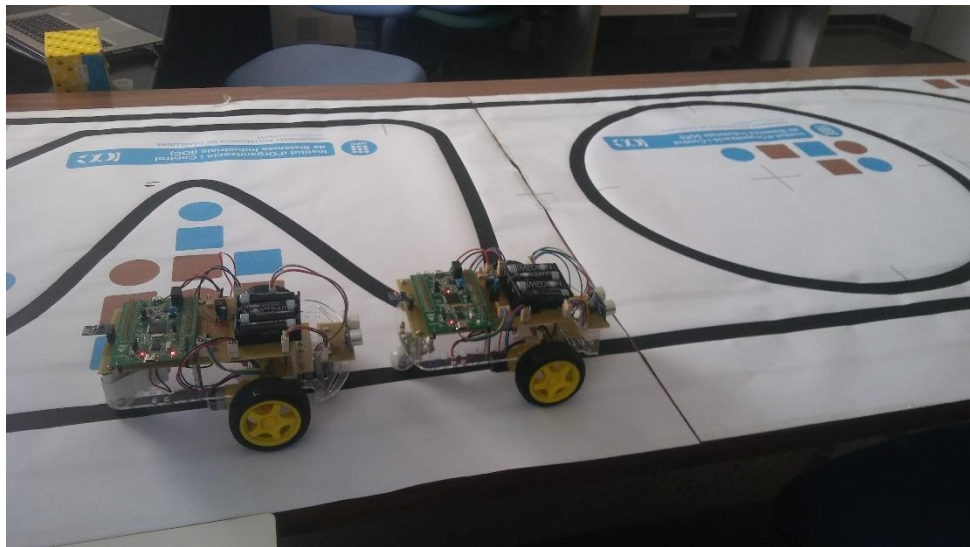
To test if vehicles are working in the desired conditions, it has been taken a circuit designed externally in the same department, whose length is higher and curvature lower than the ones available in the laboratory. It is formed by two parts, with a DIN A0 format each one. Both have been designed to join them and create the entire circuit.



(Figure 7-2) Left side of the circuit. DIN A0 format. (Dòria, 2017)



(Figure 7-3) Right side of the circuit. DIN A0 format. (Dòria, 2017)



(Figure 7-4) Image taken during vehicles testing over the mentioned circuit.

## Conclusions

Once the position controller is implemented in vehicles, it is important to conclude this project with the final structure that 2-wheel robots had with all modifications made during this study. This task will also facilitate work for students or professionals interested in developing the project.

In the moment of ending the project, there are two vehicles available, *vehicle 1* and *vehicle 2*, which have the same program debugged in its DSP. Each vehicle has its own characteristics, all considered in the common code. This firmware includes all work made before by project degree students and it incorporates the upgrades explained in this final degree project.

The objectives have been accomplished successfully so experiments and theory have considerable similarity. By the way, during the project's fulfilment it has been found that there are several options to develop all work made. A possible and important upgrade to continue the project would be the introduction of communication between vehicles. Consequently, the controller implemented (Adaptive Cruise Control) could be changed to a Cooperative Adaptive Cruise Control, whose error tends always to zero. Another possible upgrade would be to study and implement a system to allow coherent distance measures between vehicles in curves. A possibility would be to implement one more Ultrasonic Sensor in every vehicle with a specific angle

## Acknowledgements

Firstly, I would like to mention that I really appreciate all help received from my tutors Arnau Dòria Cerezo and Víctor Repecho del Corral. Both have been giving me advice and teaching me to overcome all theoretical and practical problems found during the realization of the project. Thanks to them, I have had the possibility of learning a lot in different fields of technology: automatic control, mathematics, electronics, etc. However, I would also like to thank their patience shown while teaching and the scientific non-academic acknowledgment that they have contributed to me, I mean, tricks to facilitate future tasks and problems, which I will probably find in a non-remote future. I would summarize this 4 months in academic, professional and personal learning.

Also I would like to appreciate all previous work done by Ivan Prats Martinho, Antoni Riera Seguí and Albert Costa Ruiz, whose Final Degree Projects have been really helpful to achieve this one. They did a great effort explaining with details all their work. In the same way, I would like to say thanks to the members of the IOC, for the assistance provided while needed.

Finally, I cannot forget to express my gratitude for all support received from my family and friends, not just during the project. I am sure that without their backing, I would not have had enough forces to end the degree. I will not get tired of saying thanks for everything you have done for me.

## Bibliography

- Costa, A. (2017). *Design of controllers and its implementation for a line tracker vehicle*. Barcelona: UPC, Final Degree Project.
- Dòria-Cerezo, A. (2017). *Cooperative Adaptive Cruise Control*. IOC. Barcelona: IOC, UPC. Technical Report.
- Elecfreaks.com. (n.d.). Ultrasonic Ranging Module HC - SR04 Datasheet.
- Espressif. (2017). ESP8266EX Datasheet.
- Leantec. (n.d.). Hoja de datos Sensor LRE-F22.
- Prats, I. (2016). *Control design and implementation for a line tracker vehicle*. Barcelona: UPC, Final Degree Project.
- Riera, A. (2016). *Disseny i implementació d'un Sistema de comunicacions Wi-Fi per a una xarxa de vehicles autònoms*. Barcelona: UPC, Final Degree Project.
- STMicroelectronics. (2000, January). L298 DUAL FULL-BRIDGE DRIVER Datasheet.
- STMicroelectronics. (2014, March). LM217, LM317. Datasheet.
- STMicroelectronics. (2017, May). UM1472 User Manual. Discovery kit with STM32F407VG MCU.

## Technical Concepts

In this last section of the report, technical concepts whose meaning has been obviated during the writing of the project are clearly explained to allow a good comprehension of the entire document.

---

<sup>1</sup>**Firmware:** Type of software that provides control, monitoring and data manipulation of engineered products and systems.

<sup>2</sup>**Driver:** electrical circuit or other electronic component used to control another circuit or component. Usually used to regulate current flowing through a circuit or to control other factors such as other components, some devices in the circuit.

<sup>3</sup>**Ultrasonic signal:** Sound waves with frequencies higher than the upper audible limit of human hearing.

<sup>4</sup>**DC motor:** Any of a class of rotary electrical machines that converts direct current electrical energy into mechanical energy.

<sup>5</sup>**MAC address (Media Access Control address):** unique identifier assigned to network interfaces for communications at the data link layer of a network segment.

<sup>6</sup>**PCB:** Printed circuit board, which includes connections between electronical components inside.

<sup>7</sup>**UART:** computer hardware device for asynchronous serial communication in which the data format and transmission speeds are configurable.

<sup>8</sup>**Block diagram:** Diagram of a system in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks.

<sup>9</sup>**Transfer Function:** Relation between input and output variables of a system.  $tf = \frac{Output}{Input}$ .

<sup>10</sup>**IP address (Internet Protocol address):** An identifier assigned to each computer and other device to connect to a TCP/IP network that is used to locate and identify the node in communications with other nodes on the network.

<sup>11</sup>**Baud rate:** Number of symbol changes, waveform changes, or signalling events, across the transmission medium per time unit using a digitally modulated signal or a line code.

<sup>12</sup>**AT Command:** Language, which allows communication between the Wi-Fi module and other components like a PCB or a personal computer.